[SK-AI Course: Deep Learning Basics]

Convolutional Neural Networks

Part 2: Cases Studies

Heung-Il Suk

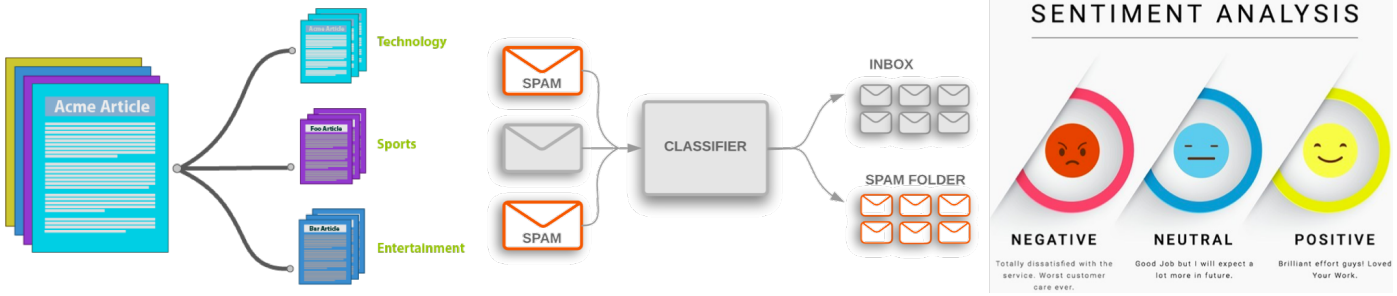
hisuk@korea.ac.kr

http://milab.korea.ac.kr

Department of Brain and Cognitive Engineering,
Korea University

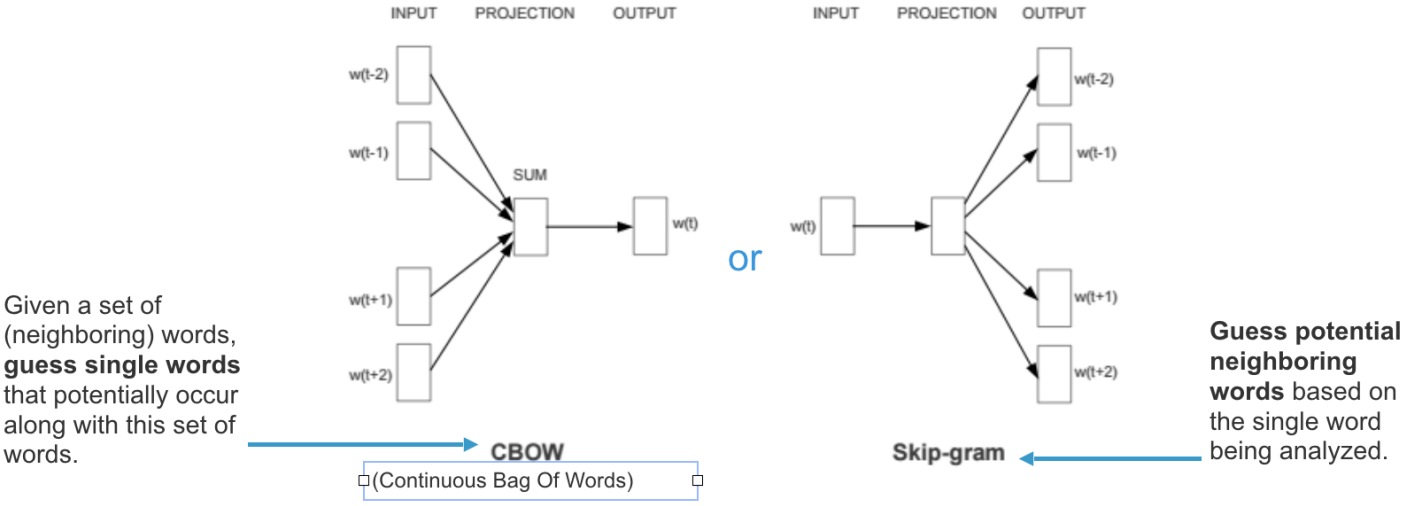
CNN for Text Classification

to automatically classify the text documents into one or more predefined categories



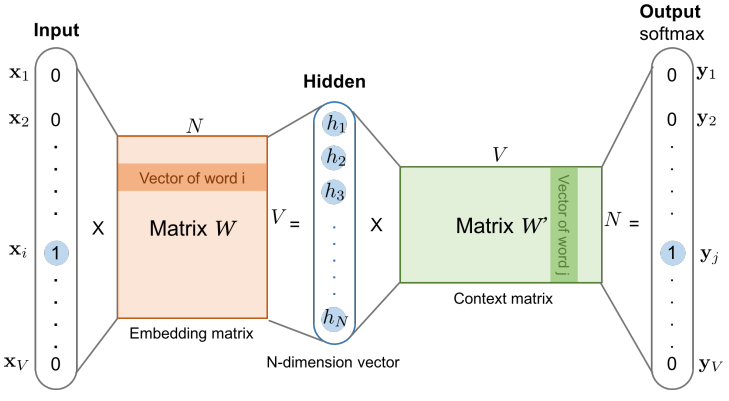
Word Embedding / Word2Vec

: to generate vector representations of words that carry semantic meanings for further NLP tasks



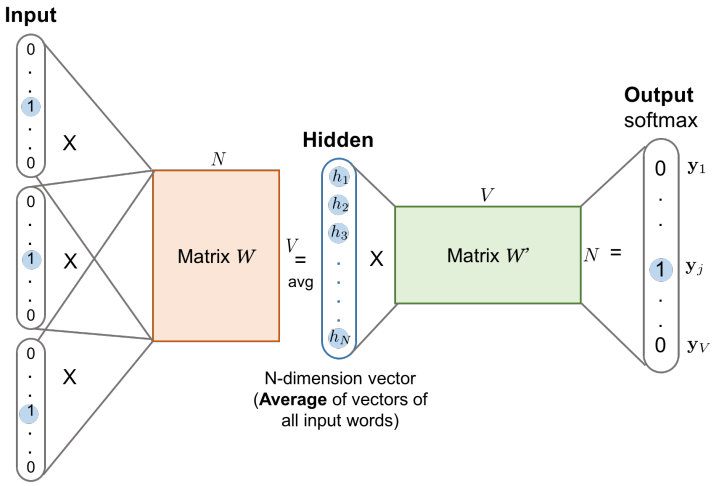
https://towardsdatascience.com/an-implementation-guide-to-word2vec-using-numpy-and-google-sheets-13445eebd281

Skip-Gram

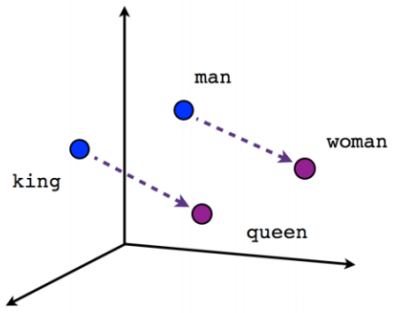


- works well with small amount of the training data
- represents well even rare words or phrases

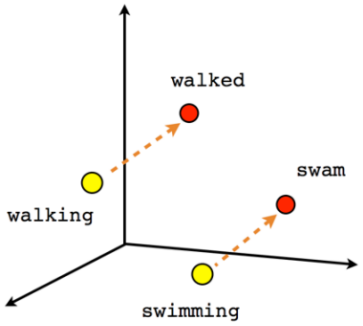
CBOW



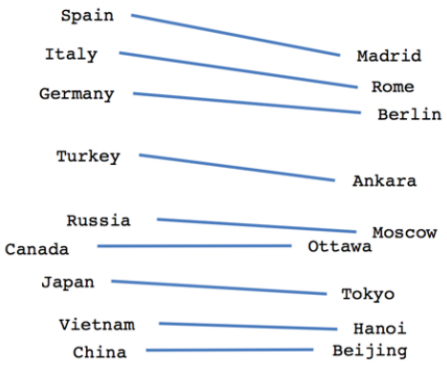
- several times faster to train than the skip-gram
- slightly better accuracy for the frequent words



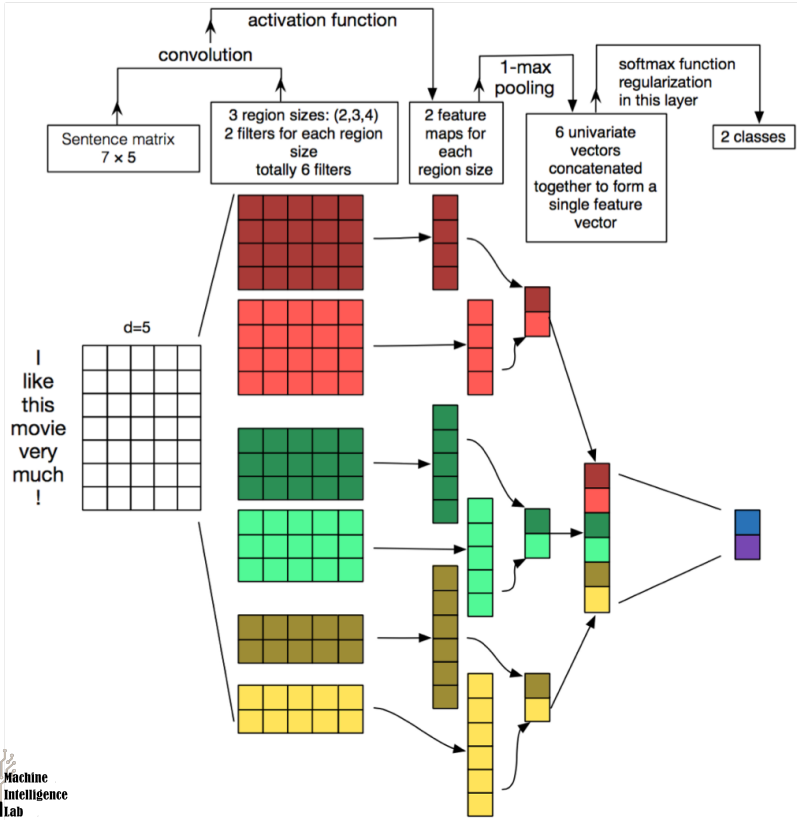
Male-Female

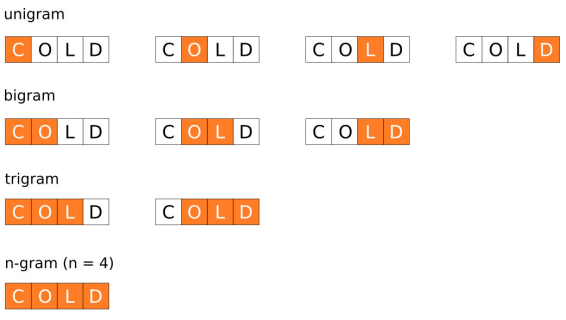


Verb tense



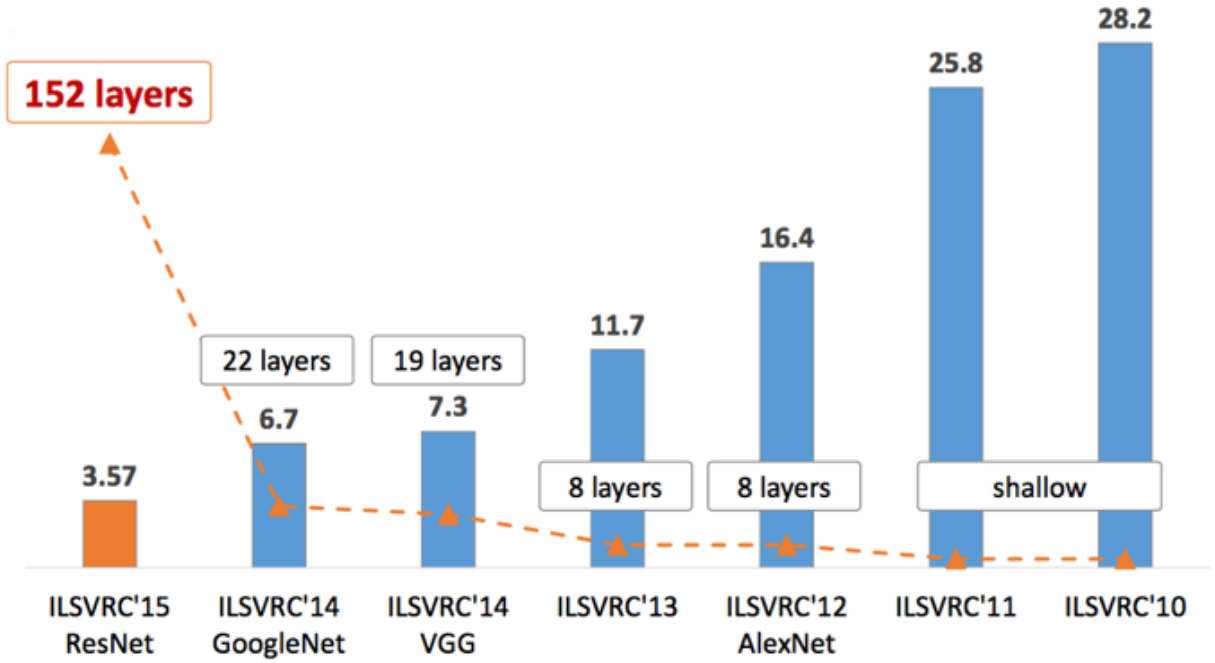
Country-Capital



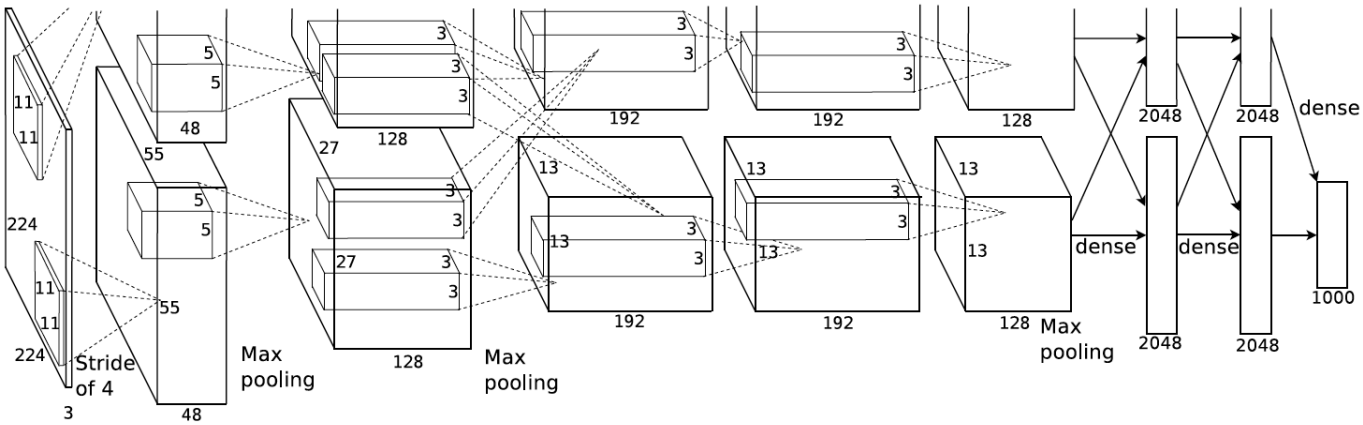


CNN Architectures

Case Studies



AlexNet [Krizhevsky *et al.*, 2014]



- Use ReLU instead of tanh to add non-linearity. (accelerating the speed by 6 times at the same accuracy)
  - ReLU: after every convolution and dense layers
- Use dropout instead of regularization to deal with overfitting. (doubled training time with dropout rate of 0.5)
  - Dropout: before the first and the second dense layers
- Overlap pooling to reduce the size of network.

Local Response Normalization (LRN)

- "lateral inhibition" in neurobiology
  - the capacity of an excited neuron to subdue its neighbors
  - to create a contrast in that area, hence increasing the sensory perception
- Because ReLU neurons have unbounded activations; need LRN to normalize that
- Dampen the responses that are uniformly large in any given local neighborhood

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} \left(a^i_{x,y}\right)^2\right)^\beta}$$

- $a^i_{x,y}$: activity by kernel i at (x, y)
- sum runs over n "adjacent" kernel maps at the same spatial position
- N: total number of kernels in the layer
- hyperparameters: $k = 2$, $n = 5$, $\alpha = 10^{-4}$, $\beta = 0.75$

in practice: either normalize within the same channel or across channels

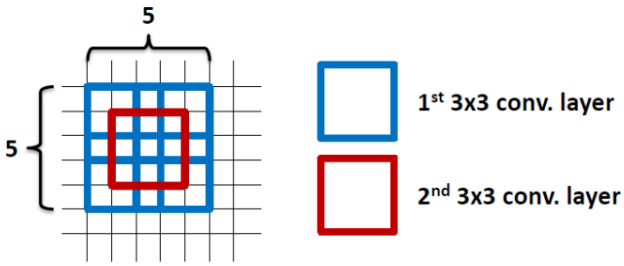
- Copy convolution layers into different GPUs; Distribute the dense layers into different GPUs.
- Feed one batch of training data into convolutional layers for every GPU (Data Parallel).
- Feed the results of convolutional layers into the distributed fully connected layers batch by batch (Model Parallel) When the last step is done for every GPU.
- Backpropogate gradients batch by batch and synchronize the weights of the convolutional layers.

- SGD with learning rate 0.01, momentum 0.9 and weight decay 0.0005
- Learning rate is divided by 10 once the the accuracy plateaus.
- The leaning rate is decreased 3 times during the training process.

https://medium.com/@smallfishbigsea/a-walk-through-of-alexnet-6cbd137a5637

VGGNet [Simonyan and Zisserman, 2015]

https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11



use of multiple layers with smaller-sized kernels for the same coverage: enhancing non-linearity

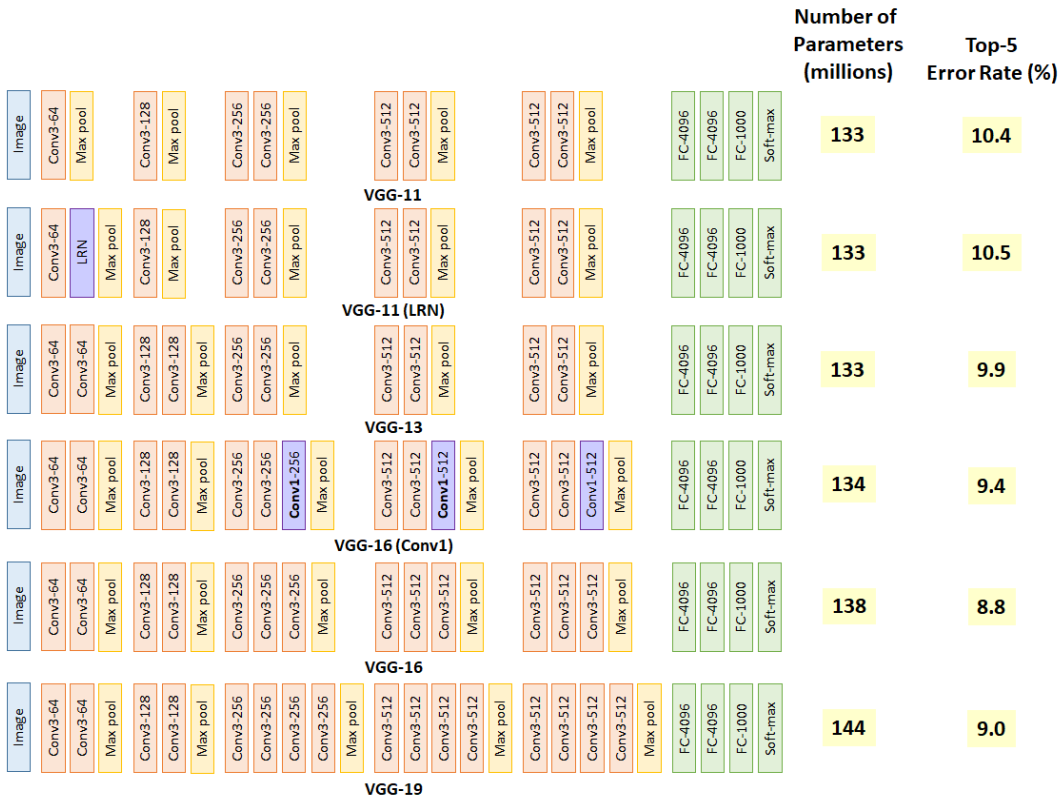
1 layer of 11×11 filter, number of parameters = **11×11=121**
5 layer of 3×3 filter, number of parameters = **3×3×5=45**
Number of parameters is reduced by 63%

1 layer of 7×7 filter, number of parameters = **7×7=49**
3 layers of 3×3 filters, number of parameters = **3×3×3=27**
Number of parameters is reduced by 45%

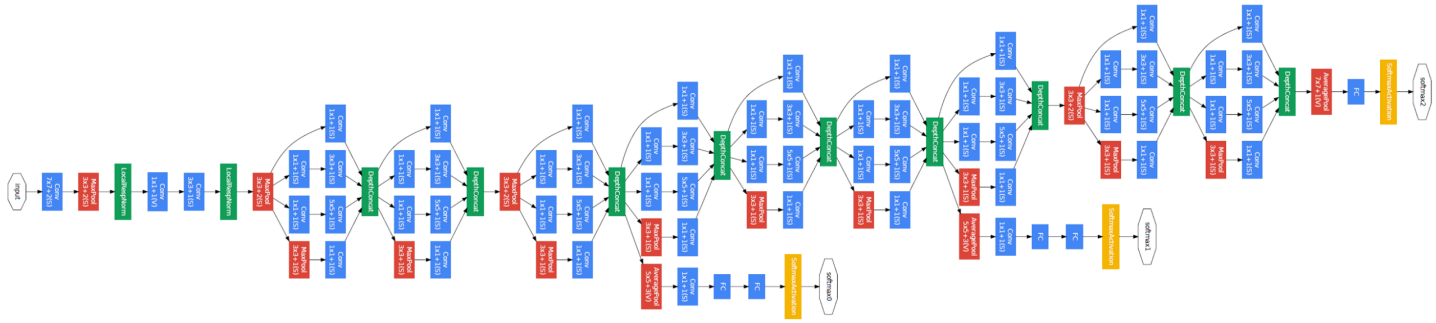
By using **1 layer of 5×5 filter**, number of parameters = **5×5=25**
By using **2 layers of 3×3 filters**, number of parameters = **3×3+3×3=18**
Number of parameters is reduced by 28%

Ablation Study:

Different VGG layer structures using single scale (256) evaluation



GoogLeNet / Inception-v1 [Szegedy *et al.*, 2015]

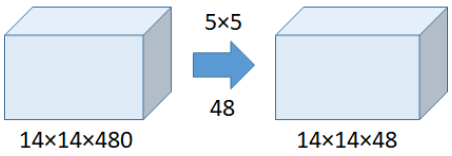


- Inception modules with 1×1 convolution
- Auxiliary Classifiers for Training
  - intermediate softmax branches at the middle (for training only)
  - for combating gradient vanishing problem and also providing regularization

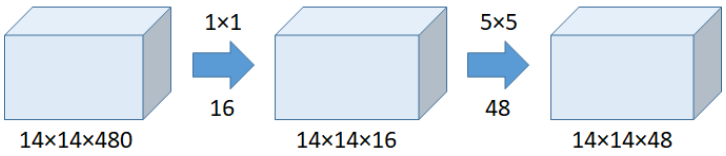
1×1 convolution

- As a dimension reduction module to reduce the computation
- By reducing the computation bottleneck, depth and width can be increased.

c.f.) NIN: introducing more non-linearity to increase the representational power of the network

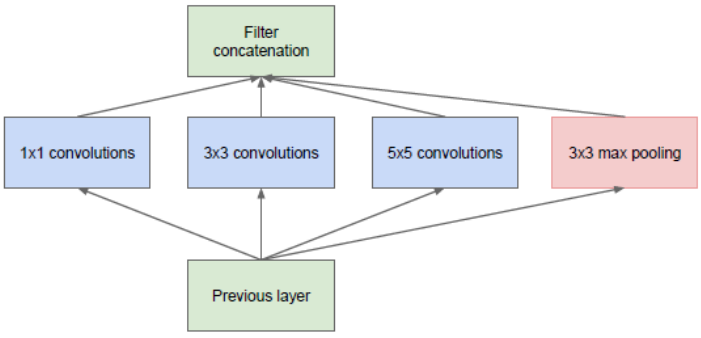


Without the Use of 1×1 Convolution



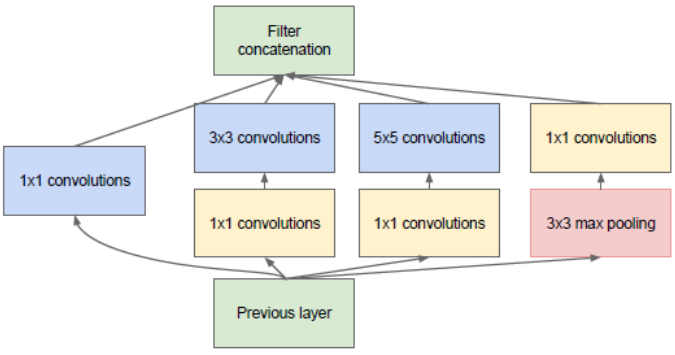
With the Use of 1×1 Convolution

Inception Module



(a) Inception module, naïve version

Naïve version (Without 1×1 Convolution)

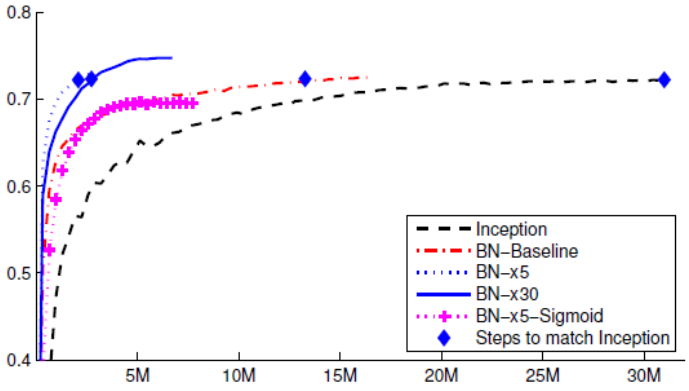


(b) Inception module with dimensionality reduction

With the Use of 1×1 Convolution

BN-Inception / Inception-v2

Applying BN to GoogLeNet / Inception-v1



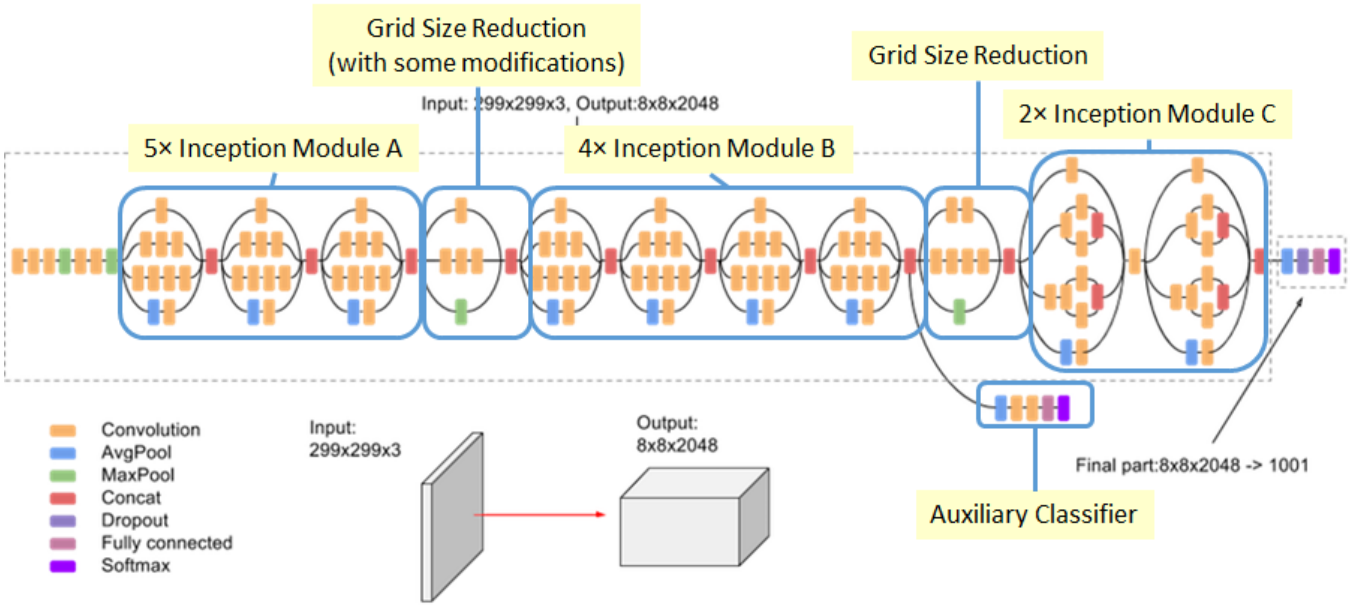
- Inception: Inception-v1 without BN
- BN-Baseline: Inception with BN
- BN-x5: Initial learning rate is increased by a factor of 5 to 0.0075
- BN-x30: Initial learning rate is increased by a factor of 30 to 0.045
- BN-x5-Sigmoid: BN-x5 but with sigmoid

https://medium.com/@sh.tsang/review-batch-normalization-inception-v2-bn-inception-the-2nd-to-surpass-human-level-18e2d0f56651

Comparison with the SOTA approaches (winner in ILSVRC 2014)

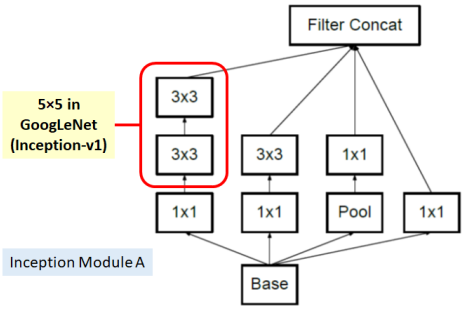
	Model	Resolution	Crops	Models	Top-1 error	Top-5 error
Inception-v1	GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image by Baidu	Deep Image low-res	256	-	1	-	7.96%
	Deep Image high-res	512	-	1	24.88	7.42%
	Deep Image ensemble	up to 512	-	-	-	5.98%
PReLU-Net	MSRA multicrop	up to 480	-	-	-	5.71%
	MSRA ensemble	up to 480	-	-	-	4.94%*
Inception-v2 / BN-Inception	BN-Inception single crop	224	1	1	25.2%	7.82%
	BN-Inception multicrop	224	144	1	21.99%	5.82%
	BN-Inception ensemble	224	144	6	20.1%	**4.82%***

Inception-v3 [Szegedy *et al.*, 2016]

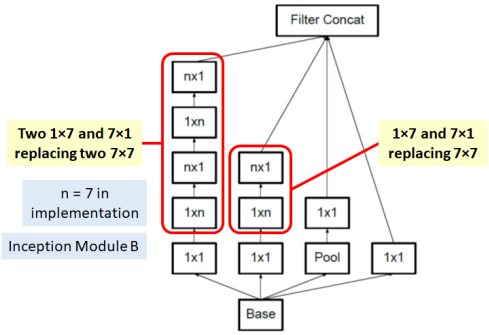


https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

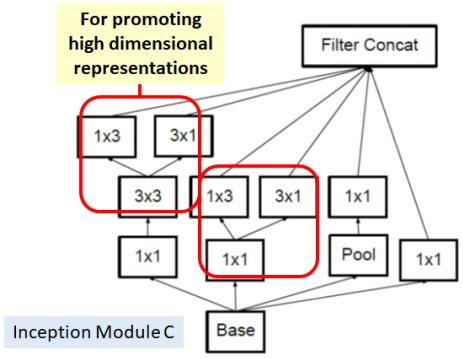
Factorization



Inception module A
using factorization

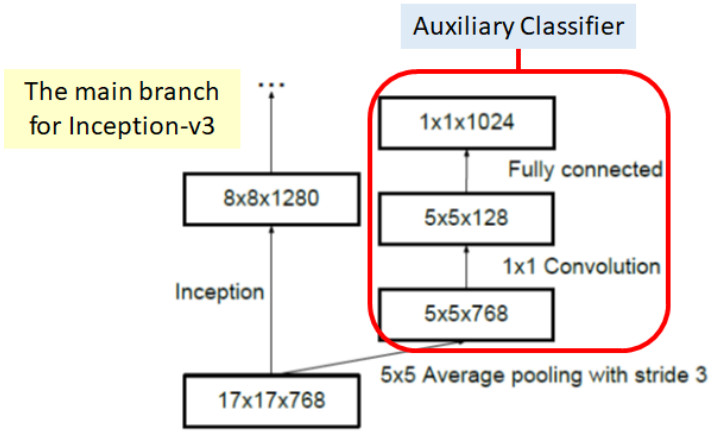


Inception module B
using asymmetric factorization



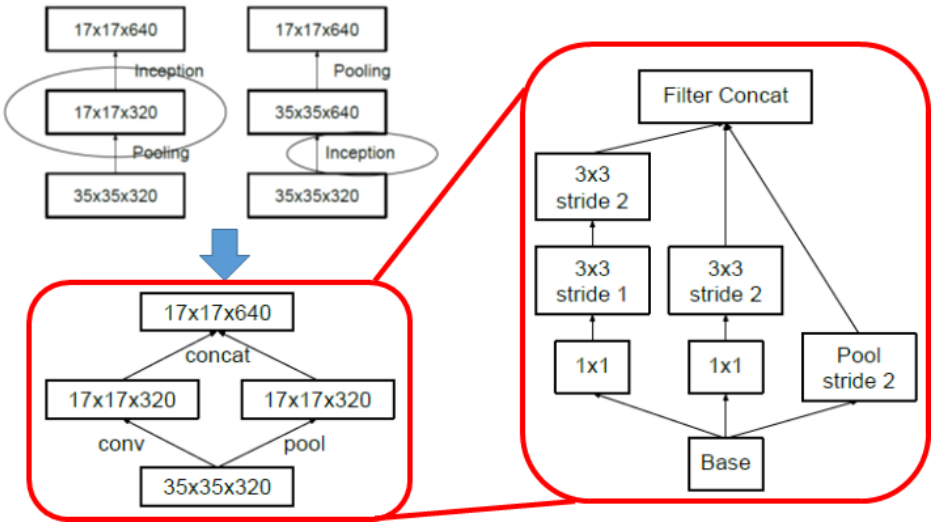
Inception module C
using asymmetric factorization

Auxiliary Classifier



Act as a regularization: only 1 auxiliary classifier with BN is used on the top of the last 17×17 layer, instead of using 2 auxiliary classifiers

Efficient Grid Size Reduction



Conventional downsizing (top-left), efficient grid size reduction (bottom-left), detailed architecture of efficient grid size reduction (right)

Label Smoothing as Regularization

$$\text{new-labels} = (1-\epsilon)\times \text{ one-hot-labels } +\epsilon/K$$

$$\text{where } \epsilon = 0.1,\ K = 1000\ (\#\text{ of classes})$$

- to prevent the largest logit from becoming much larger than all others
- a kind of dropout effect observed in classifier layer

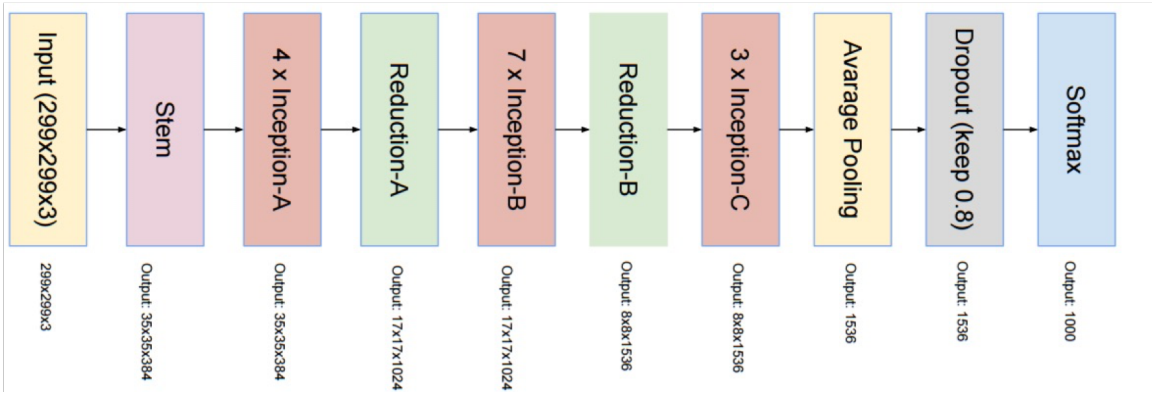
Ablation study

	Network	Top-1 Error	Top-5 Error	Cost Bn Ops
Inception-v1	GoogLeNet [20]	29%	9.2%	1.5
	BN-GoogLeNet	26.8%	-	**1.5**
Inception-v2	BN-Inception [7]	25.2%	7.8	2.0
Inception-v3	Inception-v3-basic	23.4%	-	3.8
Inception-v3 + RMSProp	Inception-v3-rmsprop RMSProp	23.1%	6.3	3.8
Inception-v3 + RMSProp + Label Smoothing	Inception-v3-smooth Label Smoothing	22.8%	6.1	3.8
Inception-v3 + RMSProp + Label Smoothing + 7×7 Factorization	Inception-v3-fact Factorized 7×7	21.6%	5.8	4.8
Inception-v3 + RMSProp + Label Smoothing + 7×7 Factorization + Auxiliary Classifier	Inception-v3 BN-auxiliary	**21.2%**	**5.6%**	4.8

Comparison with the SOTA approaches

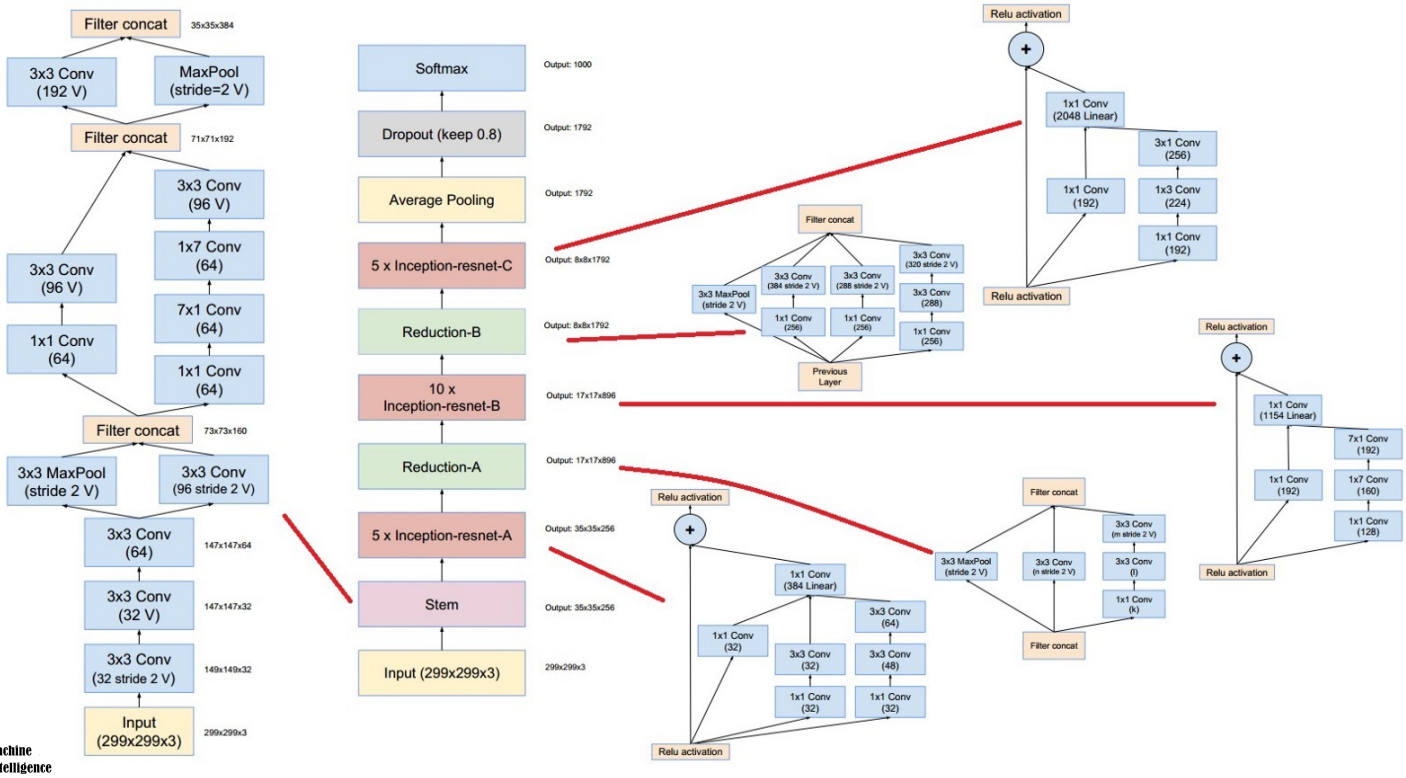
	Network	Crops Evaluated	Top-1 Error	Top-5 Error
Inception-v1	GoogLeNet [20]	10	-	9.15%
	GoogLeNet [20]	144	-	7.89%
VGGNet	VGG [18]	-	24.4%	6.8%
Inception-v2	BN-Inception [7]	144	22%	5.82%
PReLU-Net	PReLU [6]	10	24.27%	7.38%
	PReLU [6]	-	21.59%	5.71%
Inception-v3 with different techniques	Inception-v3	12	19.47%	4.48%
	Inception-v3	144	**18.77%**	**4.2%**

Inception-V4 [Szegedy *et al.*, 2017]



A more uniform simplified architecture and more inception modules than Inception-v3 a shortcut connection at the left of each inception module allowing to go deeper as in ResNet

https://towardsdatascience.com/review-inception-v4-evolved-from-googlenet-merged-with-resnet-idea-image-classification-5e8c339d18bc



Network	Top-1 Error	Top-5 Error
BN-Inception (Ioffe and Szegedy 2015)	25.2%	7.8%
Inception-v3 (Szegedy et al. 2015b)	21.2%	5.6%
Inception-ResNet-v1	21.3%	5.5%
Inception-v4	20.0%	5.0%
Inception-ResNet-v2	19.9%	4.9%

Single-Crop Single-Model Results

Network	Crops	Top-1 Error	Top-5 Error
ResNet (He et al. 2015)	10	25.2%	7.8%
Inception-v3 (Szegedy et al. 2015b)	12	19.8%	4.6%
Inception-ResNet-v1	12	19.8%	4.6%
Inception-v4	12	18.7%	4.2%
Inception-ResNet-v2	12	18.7%	4.1%

10/12-Crop Single-Model Results

Network	Crops	Top-1 Error	Top-5 Error
Inception-v3 (Szegedy et al. 2015b)	144	18.9%	4.3%
Inception-ResNet-v1	144	18.8%	4.3%
Inception-v4	144	17.7%	3.8%
Inception-ResNet-v2	144	17.8%	3.7%

144-Crop Single-Model Results

Winner in ILSVRC 2015

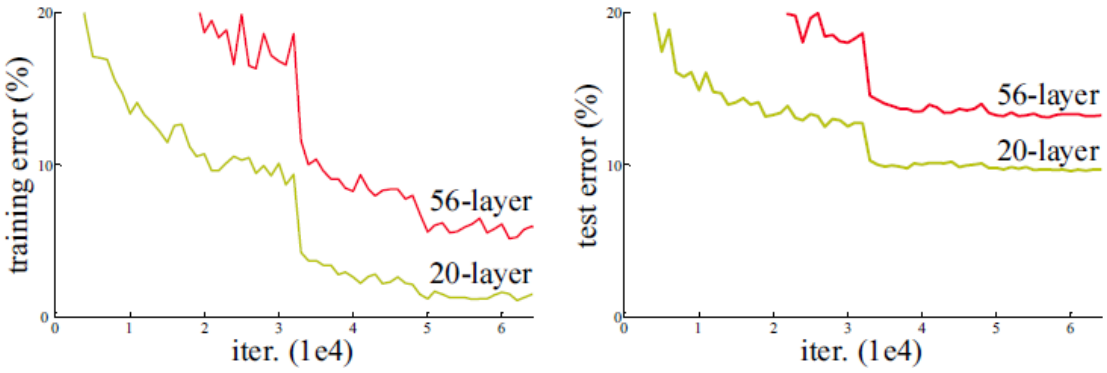
1s Runner Up in ILSVRC 2015

Network	N	Top-1 Error	Top-5 Error
ResNet (He et al. 2015)	6	N/A	3.6%
Inception-v3 (Szegedy et al. 2015b)	4	17.3%	3.6%
Inception-v4(+Residual)	4	16.4%	3.1%

144-Crop N-Model Results

ResNet [He *et al.*, 2015]

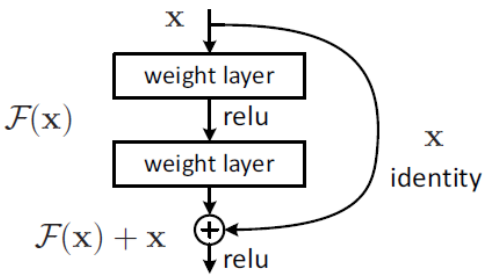
Vanishing/exploding problem



the deeper a network, the lower the performance in plain networks

https://towardsdatascience.com/review-resnet-winner-of-ilsvrc-2015-image-classification-localization-detection-e39402bfa5d8

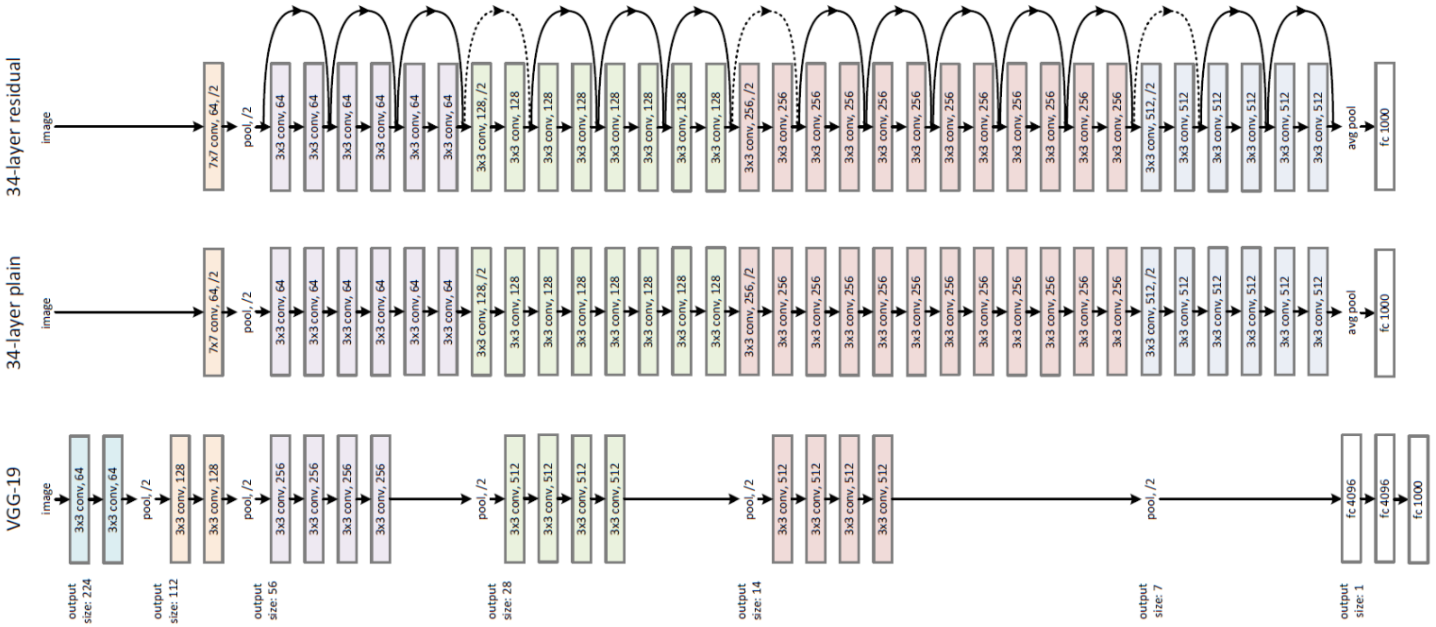
Skip/shortcut connection as a rescue



- add the input $\mathbf{x}$ to the output after a few weight layers
- weight layers: to learn a kind of residual mapping

$$\mathcal{F}(\mathbf{x}) = \mathcal{H}(\mathbf{x}) - \mathbf{x}$$

Even if there is vanishing gradient for the weight layers,
we always still have the identity $\mathbf{x}$ to transfer back to earlier layers



When the input dimension $\mathbf{x}$ is smaller than the output dimensions $\mathcal{F}(\mathbf{x})$

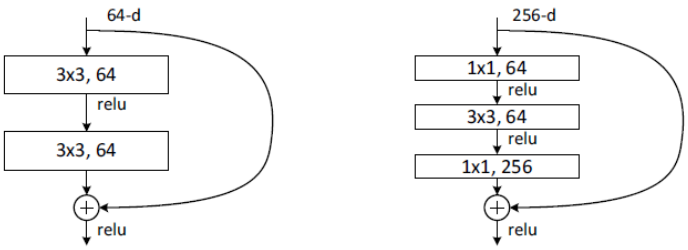
(1) Shortcut performs identity mapping, with extra zero padding for increasing dimensions.
Thus, no extra parameters.

(2) The projection shortcut is used for increasing dimensions only, the other shortcuts are identity. Extra parameters are needed.

(3) All shortcuts are projections. Extra parameters are more than that of (2)

Bottleneck design

- Since the network is very deep now, the time complexity is high.
- A bottleneck design is used to reduce the complexity.

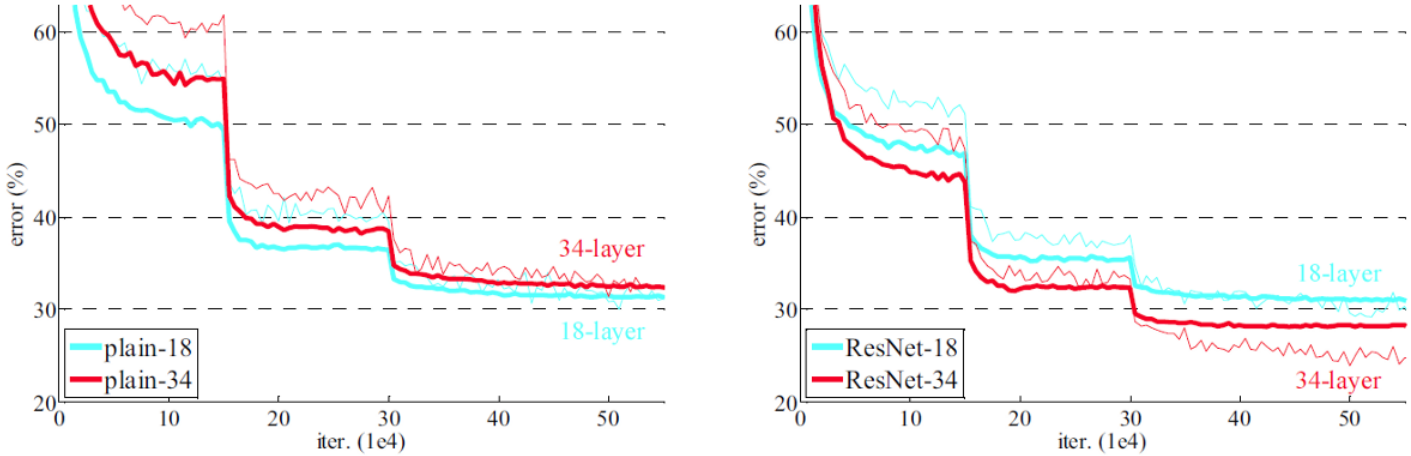


(left) basic block, (right) bottleneck design

- 1×1 conv
  - *c.f.*, Network In Network and GoogLeNet (Inception-v1)
  - reducing the number of connections (parameters) while not degrading the performance of the network so much
- 34-layer ResNet $\rightarrow$ 50-layer ResNet
- Deeper network with the bottleneck design
  - ResNet-101 and ResNet-152

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

VGG-16/19 has 15.3/19.6 billion FLOPS. ResNet-152 still has lower complexity than VGG-16/19!!!



Validation error: 18- and 34-layer plain network (Left), 18- and 34-layer ResNet (right)

Image classification: CIFAR-10 ILSVRC

- Batch Normalization (from Inception-v2) is used after each conv.

(a) 10-crop testing[1]

(b) Fully convolutional form with averaging the scores at multiple scales {224, 256, 384, 480, 640}

(c) ensemble boosting with 6 models

(a)

model	top-1 err.	top-5 err.
VGG-16 [40]	28.07	9.33
GoogLeNet [43]	-	9.15
PReLU-net [12]	24.27	7.38
plain-34	28.54	10.02
ResNet-34 A	25.03	7.76
ResNet-34 B	24.52	7.46
ResNet-34 C	24.19	7.40
ResNet-50	22.85	6.71
ResNet-101	21.75	6.05
ResNet-152	**21.43**	**5.71**

(a) + (b)

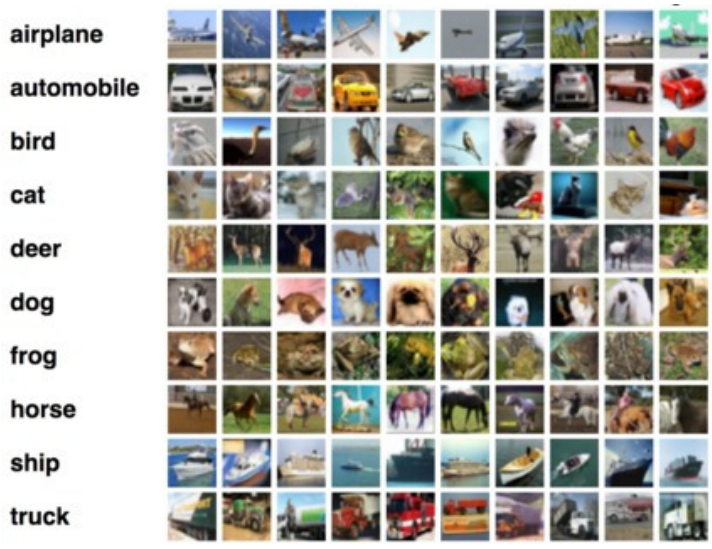
method	top-1 err.	top-5 err.
VGG [40] (ILSVRC'14)	-	8.43†
GoogLeNet [43] (ILSVRC'14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

(a) + (b) + (c)

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

[1] Four crops from each corner, as well as one from the center, and then performs a horizontal flip on each to pass 10 crops from a single test image through the network. The classification scores are then combined to determine the most likely class

Image classification: CIFAR-10



method	# layers	# params	error (%)
Maxout [9]			9.38
NIN [25]			8.81
DSN [24]			8.22
FitNet [34]	19	2.5M	8.39
Highway [41, 42]	19	2.3M	7.54 (7.72±0.16)
Highway [41, 42]	32	1.25M	8.80
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97
ResNet	110	1.7M	**6.43** (6.61±0.16)
ResNet	1202	19.4M	7.93

Comparison with the SOTA approaches: object detection

PASCAL VOC 2007/2012 mAP (%)

training data	07+12	07++12
test data	VOC 07 test	VOC 12 test
VGG-16	73.2	70.4
ResNet-101	**76.4**	**73.8**

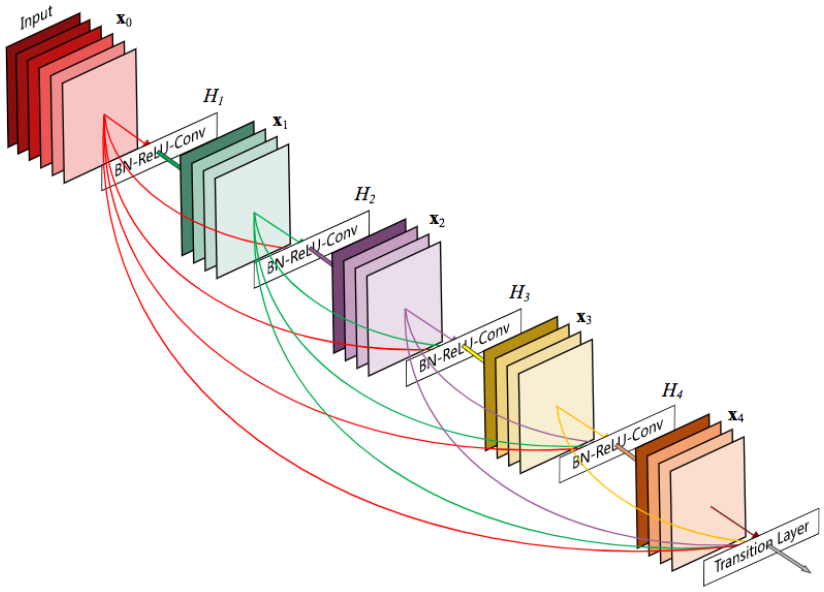
MS COCO mAP (%)

metric	mAP@.5	mAP@[.5, .95]
VGG-16	41.5	21.2
ResNet-101	**48.4**	**27.2**

- By adopting the ResNet-101 into Faster R-CNN, ResNet obtains better performance than VGG-16 by large margin.
- ResNet finally won the 1st places on ImageNet Detection, Localization, COCO Detection and COCO Segmentation!!!

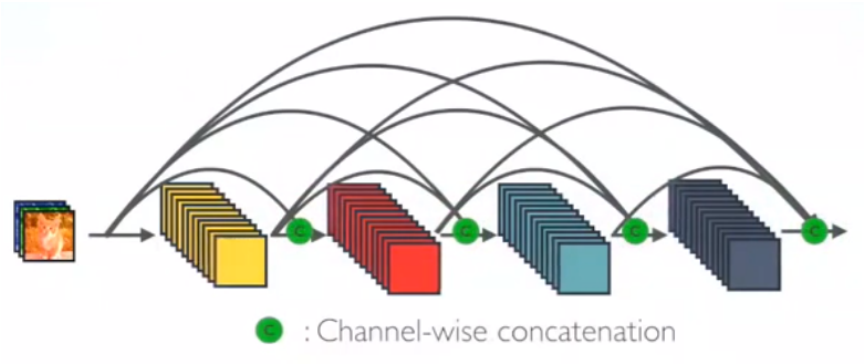
mAP: mean Average Precision (https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)

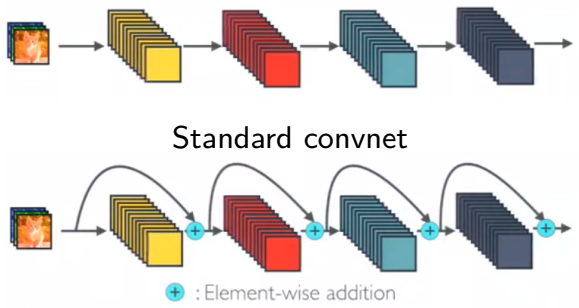
DenseNet [Huang *et al.*, 2017]



https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

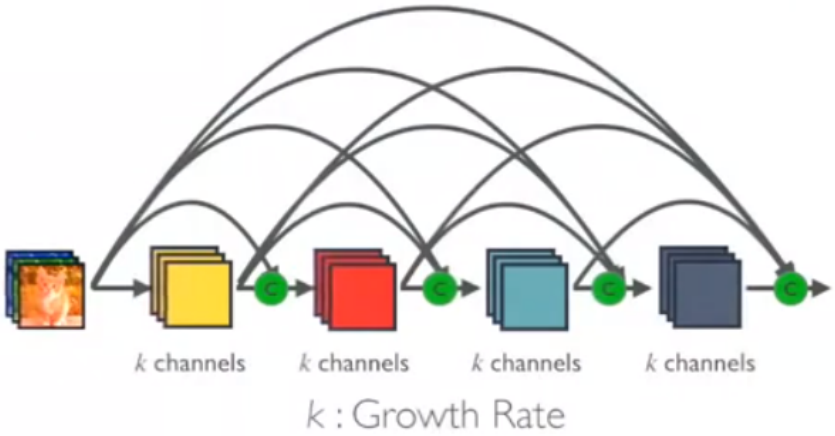
Dense Block





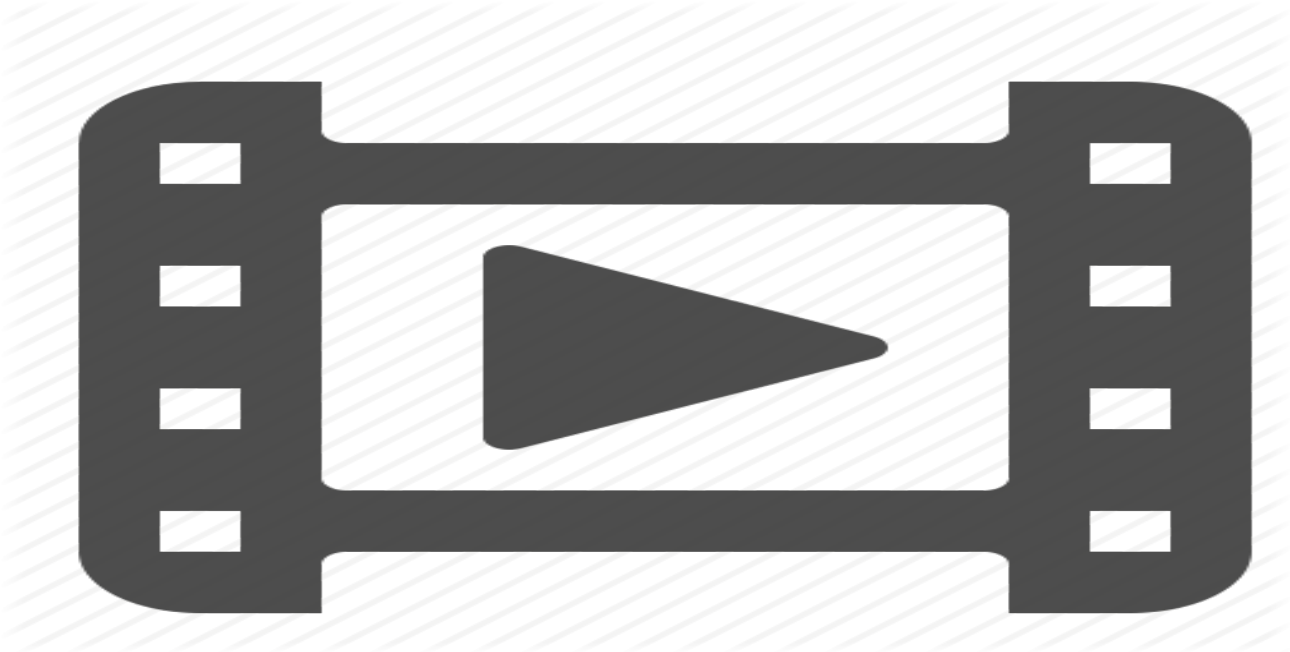
ResNet

- each layer obtains additional inputs from all preceding layers
- passes on its own feature-maps to all subsequent layers
- Each layer is receiving a "collective knowledge" from all preceding layers.

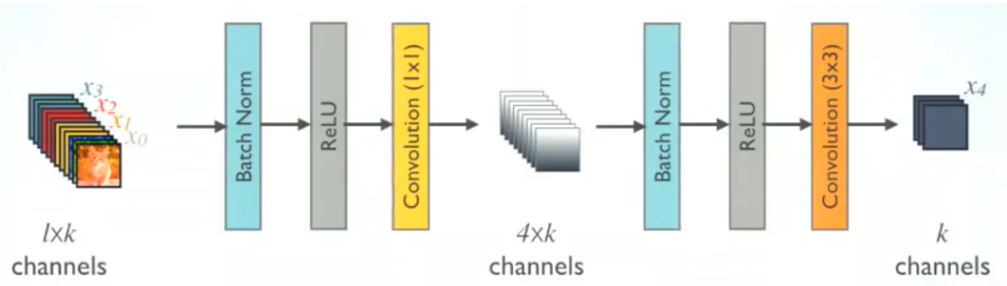


- Since each layer receives feature maps from all preceding layers, network can be thinner and compact, *i.e.*, number of channels can be fewer.
- The growth rate k is the additional number of channels for each layer.

Basic DenseNet Composition Layer

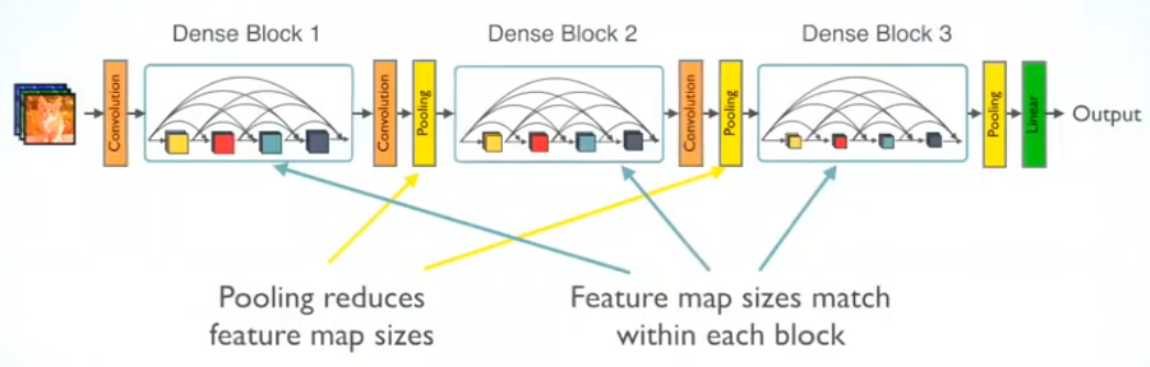


DenseNet-B (Bottleneck Layers)



- To reduce the model complexity and size, BN-ReLU-1×1 Conv before BN-ReLU-3×3 Conv

Multiple Dense Blocks with Transition Layers

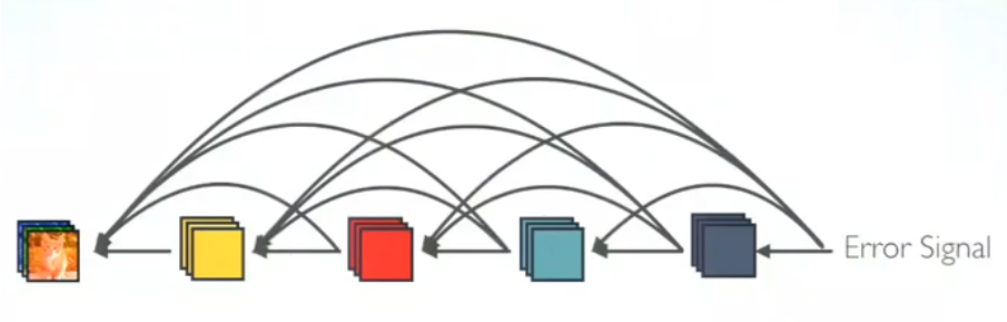


- 1×1 Conv followed by 2×2 average pooling are used as the transition layers between two contiguous dense blocks
- Feature map sizes are the same within the dense block so that they can be concatenated together easily.
- At the end of the last dense block, a global average pooling is performed and then a softmax classifier is attached.

DenseNet-BC (Further Compression)

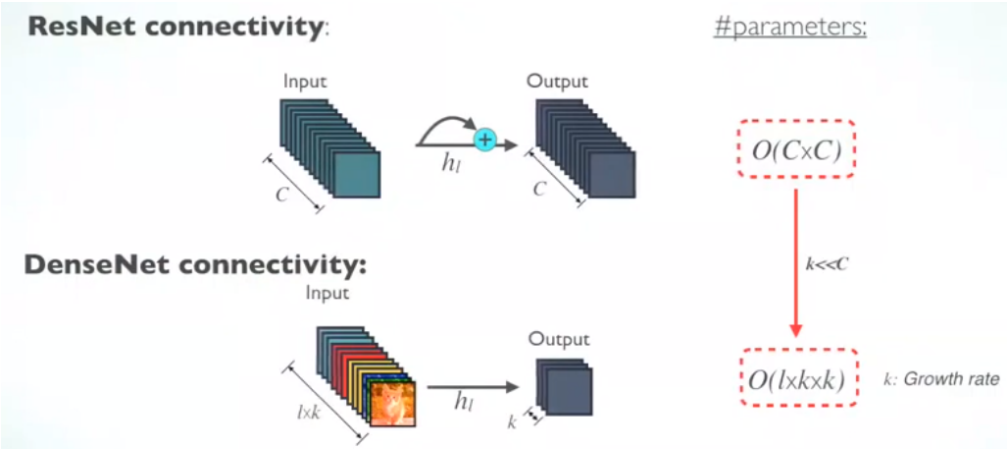
- If a dense block contains m feature maps, the transition layer generate θm output feature maps, where $0 < \theta \leq 1$ is referred to as the compression factor.
- When $\theta = 1$, the number of feature maps across transition layers remains unchanged.
- DenseNet with $\theta < 1$ is referred as DenseNet-C and $\theta = 0.5$ in the experiment.
- When both the bottleneck and transition layers with $\theta < 1$ are used, the model is referred as DenseNet-BC.

Strong Gradient Flow



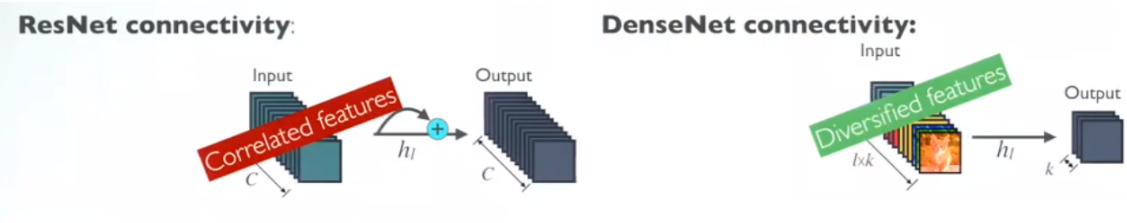
Implicit "deep supervision"

Parameter & Computational Efficiency



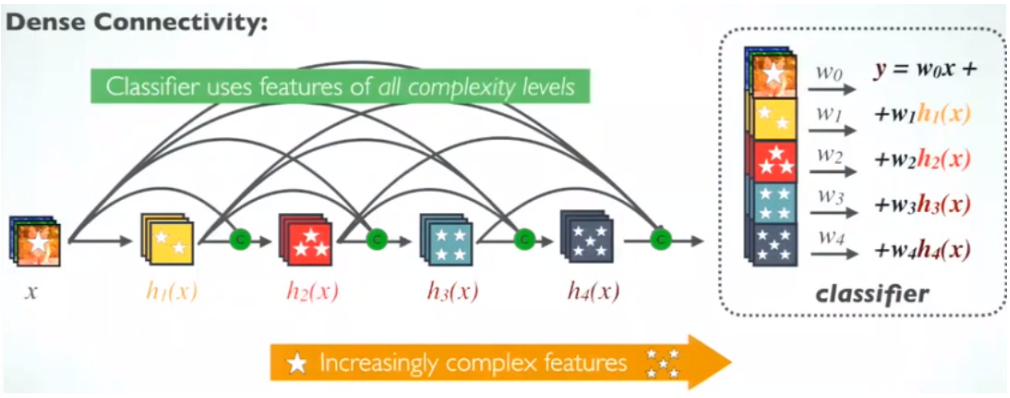
Number of parameters for ResNet and DenseNet: For each layer, number of parameters in ResNet is directly proportional to $C \times C$ while number of parameters in DenseNet is directly proportional to $l \times k \times k$.

More Diversified Features



Since each layer in DenseNet receive all preceding layers as input,
more diversified features and tends to have richer patterns.

Maintains Low Complexity Features

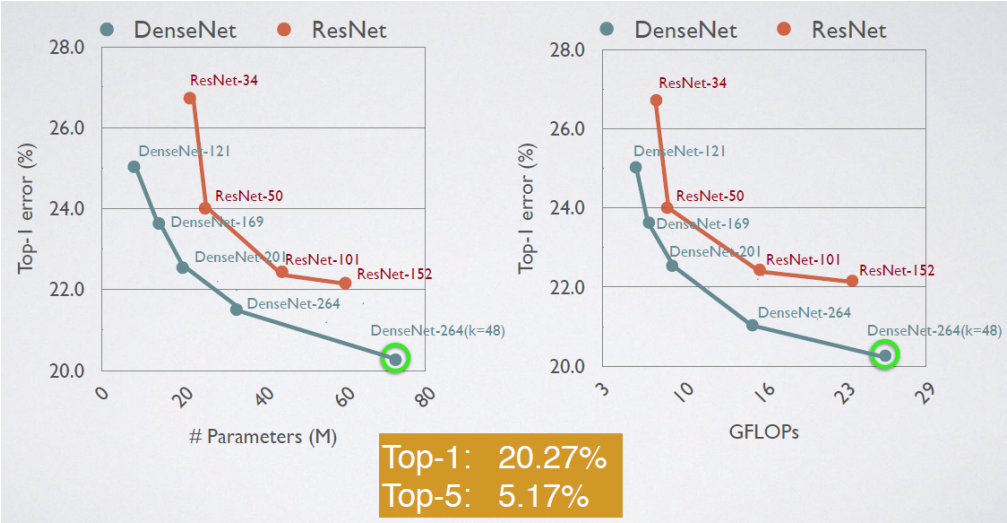


Classifier uses features of all complexity levels.
It tends to give more smooth decision boundaries.
It also explains why DenseNet performs well when training data is insufficient.

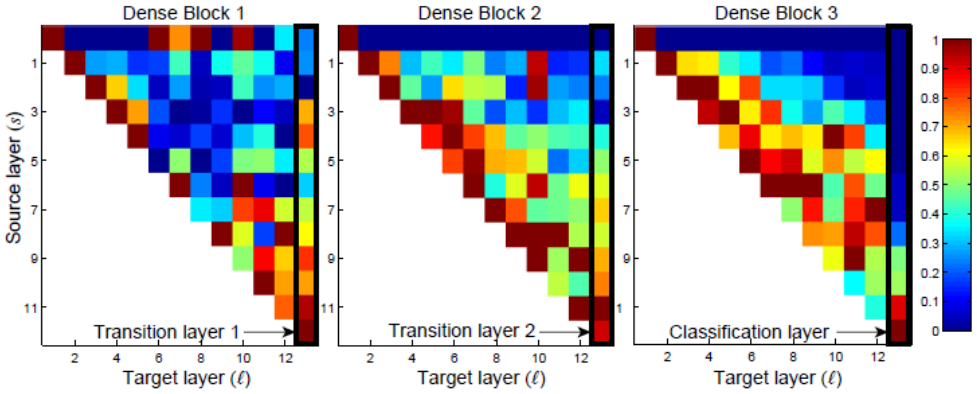
Detailed Results on CIFAR-100, '+' means data augmentation

Method	Depth	Params	C10	C10+	C100	C100+	SVHN
Network in Network [22]	-	-	10.41	8.81	35.68	-	2.35
All-CNN [32]	-	-	9.08	7.25	-	33.71	-
Deeply Supervised Net [20]	-	-	9.69	7.97	-	34.57	1.92
Highway Network [34]	-	-	-	7.72	-	32.39	-
FractalNet [17]	21	38.6M	10.18	5.22	35.34	23.30	2.01
with Dropout/Drop-path	21	38.6M	7.33	4.60	28.20	23.73	1.87
ResNet [11]	110	1.7M	-	6.61	-	-	-
ResNet (reported by [13])	110	1.7M	13.63	6.41	44.74	27.22	2.01
ResNet with Stochastic Depth [13]	110	1.7M	11.66	5.23	37.80	24.58	1.75
	1202	10.2M	-	4.91	-	-	-
Wide ResNet [42]	16	11.0M	-	4.81	-	22.07	-
	28	36.5M	-	4.17	-	20.50	-
with Dropout	16	2.7M	-	-	-	-	1.64
ResNet (pre-activation) [12]	164	1.7M	11.26*	5.46	35.58*	24.33	-
	1001	10.2M	10.56*	4.62	33.47*	22.71	-
DenseNet ($k = 12$)	40	1.0M	**7.00**	5.24	**27.55**	24.42	1.79
DenseNet ($k = 12$)	100	7.0M	**5.77**	**4.10**	**23.79**	**20.20**	1.67
DenseNet ($k = 24$)	100	27.2M	**5.83**	**3.74**	**23.42**	**19.25**	**1.59**
DenseNet-BC ($k = 12$)	100	0.8M	**5.92**	4.51	**24.15**	22.27	1.76
DenseNet-BC ($k = 24$)	250	15.3M	**5.19**	**3.62**	**19.64**	**17.60**	1.74
DenseNet-BC ($k = 40$)	190	25.6M	-	**3.46**	-	**17.18**	-

ImageNet Validation Set Results Compared with Original ResNet



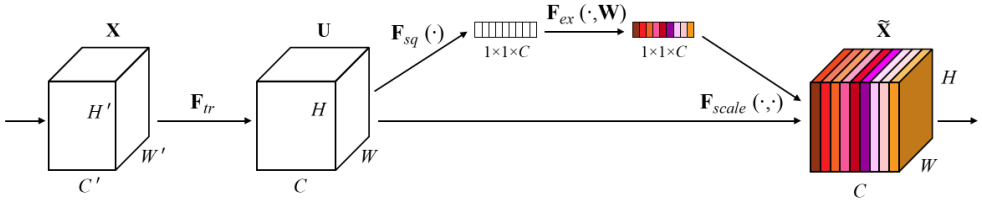
Heat map on the average absolute weights of how Target layer (l) reuses the source layer (s)



- Features extracted by very early layers are directly used by deeper layers throughout the same dense block.
- Weights of transition layers also spread their weights across all preceding layers.
- Layers within the second and third dense blocks consistently assign the least weight to the outputs of the transition layers. (The first row)
- At the final classification layer, weights seems to be a concentration towards final feature maps. Some more high-level features produce late in the network.

Squeeze-and-Excitation Networks (SENet) [Hu *et al.*, 2018]

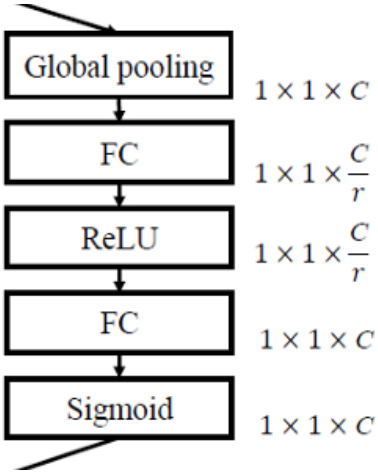
Squeeze-and-Excitation Block



- adaptively recalibrates channel-wise feature responses by explicitly modeling inter-dependencies between channels

https://towardsdatascience.com/review-senet-squeeze-and-excitation-network-winner-of-ilsvrc-2017-image-classification-a887b98b2883

Squeeze: Global Information Embedding

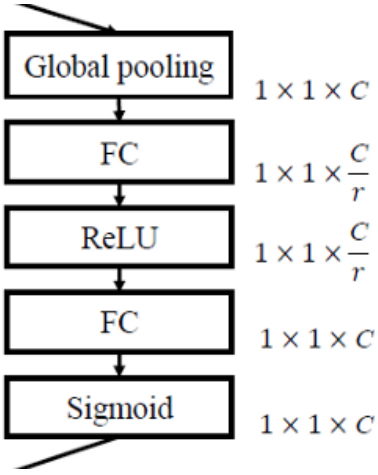


$$z_c = \mathbf{F}_{sq}(\mathbf{u}_c) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} u_c(i, j)$$

- $\mathbf{u}_c$: interpreted as a collection of the local descriptors whose statistics are expressive for the whole image
- proposed to squeeze global spatial information into a channel descriptor

Excitation: Adaptive Recalibration

$$\mathbf{s} = \mathbf{F}_{ex}(\mathbf{z}, \mathbf{W}) = \sigma\left(g\left(\mathbf{z}, \mathbf{W}\right)\right) = \sigma\left(\mathbf{W}_2 \text{ReLU}\left(\mathbf{W}_1 \mathbf{z}\right)\right)$$



- a simple **gating mechanism** using sigmoid activation σ is used
- bottleneck with two fully connected (FC) layers
  - to fully capture channel-wise dependencies
  - formed with reduction ratio r

$$\frac{2}{r} \sum_{s=1}^{S} N_s \cdot C_s^2$$

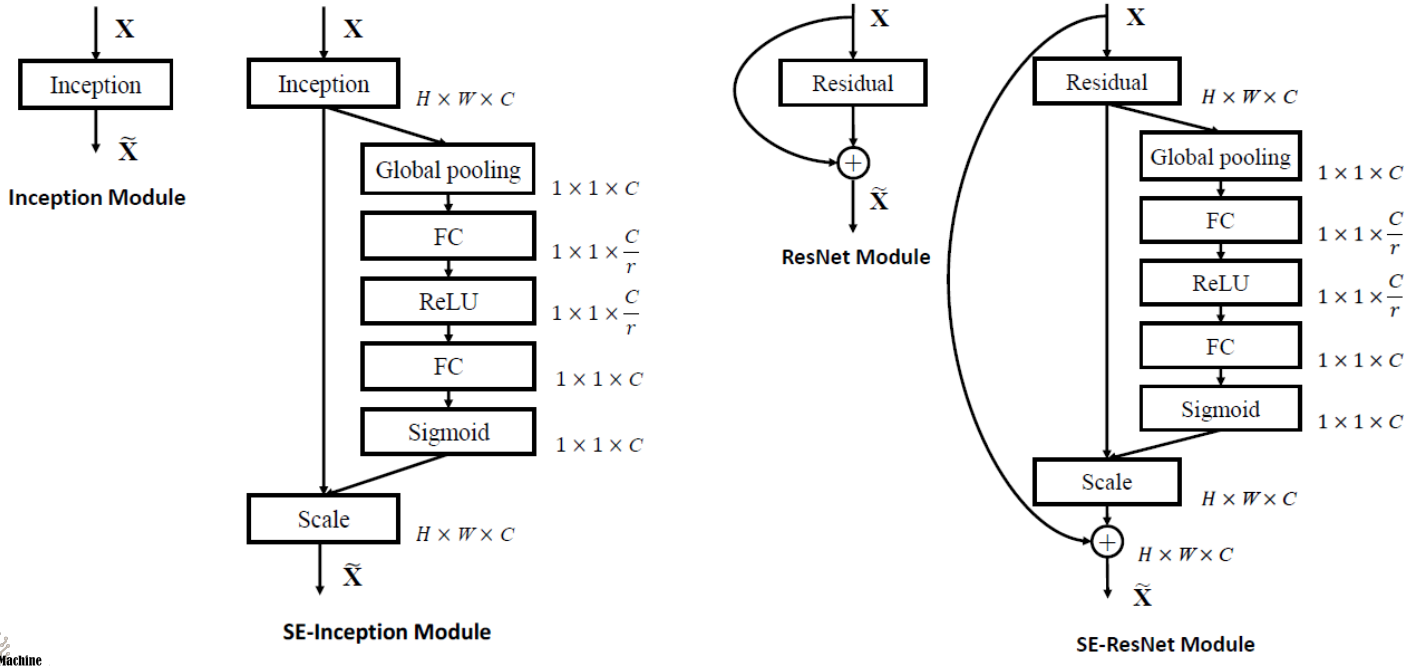
  - S: # of stages (stage: collection of blocks operating on feature maps of a common spatial dimension)
  - C_s: dimension of the output channels
  - N_s: repeated block # for stage s

Rescaling Feature Maps

$$\tilde{\mathbf{x}}_c = \mathbf{F}_{scale}\left(\mathbf{u}_c, s_c\right) = s_c \cdot \mathbf{u}_c$$

- rescaling the transformation output $\mathbf{U}$ with the activations
- The activations act as channel weights adapted to the input-specific descriptor $\mathbf{z}$.
- In this regard, SE blocks intrinsically introduce dynamics conditioned on the input, helping to boost feature discriminability.

SE-Inception & SE-ResNet



ResNet-50 (Left), SE-ResNet-50 (Middle), SE-ResNeXt-50 (32×4d) (Right)

Output size	ResNet-50	SE-ResNet-50	SE-ResNeXt-50 (32 × 4d)
112 × 112	conv, 7 × 7, 64, stride 2		
56 × 56	max pool, 3 × 3, stride 2		
	$\begin{bmatrix} \text{conv}, 1\times1, 64 \\ \text{conv}, 3\times3, 64 \\ \text{conv}, 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 64 \\ \text{conv}, 3\times3, 64 \\ \text{conv}, 1\times1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 128 \\ \text{conv}, 3\times3, 128 \quad C=32 \\ \text{conv}, 1\times1, 256 \\ fc, [16, 256] \end{bmatrix} \times 3$
28 × 28	$\begin{bmatrix} \text{conv}, 1\times1, 128 \\ \text{conv}, 3\times3, 128 \\ \text{conv}, 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv}, 1\times1, 128 \\ \text{conv}, 3\times3, 128 \\ \text{conv}, 1\times1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$	$\begin{bmatrix} \text{conv}, 1\times1, 256 \\ \text{conv}, 3\times3, 256 \quad C=32 \\ \text{conv}, 1\times1, 512 \\ fc, [32, 512] \end{bmatrix} \times 4$
14 × 14	$\begin{bmatrix} \text{conv}, 1\times1, 256 \\ \text{conv}, 3\times3, 256 \\ \text{conv}, 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv}, 1\times1, 256 \\ \text{conv}, 3\times3, 256 \\ \text{conv}, 1\times1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$	$\begin{bmatrix} \text{conv}, 1\times1, 512 \\ \text{conv}, 3\times3, 512 \quad C=32 \\ \text{conv}, 1\times1, 1024 \\ fc, [64, 1024] \end{bmatrix} \times 6$
7×7	$\begin{bmatrix} \text{conv}, 1\times1, 512 \\ \text{conv}, 3\times3, 512 \\ \text{conv}, 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 512 \\ \text{conv}, 3\times3, 512 \\ \text{conv}, 1\times1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$	$\begin{bmatrix} \text{conv}, 1\times1, 1024 \\ \text{conv}, 3\times3, 1024 \quad C=32 \\ \text{conv}, 1\times1, 2048 \\ fc, [128, 2048] \end{bmatrix} \times 3$
1 × 1	global average pool, 1000-d fc, softmax		

Single-Crop Error Rates (%) on ImageNet Validation Set

	original		re-implementation			SENet		
	top-1 err.	top-5 err.	top-1err.	top-5 err.	GFLOPs	top-1 err.	top-5 err.	GFLOPs
ResNet-50 [10]	24.7	7.8	24.80	7.48	3.86	$23.29_{(1.51)}$	$6.62_{(0.86)}$	3.87
ResNet-101 [10]	23.6	7.1	23.17	6.52	7.58	$22.38_{(0.79)}$	$6.07_{(0.45)}$	7.60
ResNet-152 [10]	23.0	6.7	22.42	6.34	11.30	$21.57_{(0.85)}$	$5.73_{(0.61)}$	11.32
ResNeXt-50 [47]	22.2	-	22.11	5.90	4.24	$21.10_{(1.01)}$	$5.49_{(0.41)}$	4.25
ResNeXt-101 [47]	21.2	5.6	21.18	5.57	7.99	$20.70_{(0.48)}$	$5.01_{(0.56)}$	8.00
VGG-16 [39]	-	-	27.02	8.81	15.47	$25.22_{(1.80)}$	$7.70_{(1.11)}$	15.48
BN-Inception [16]	25.2	7.82	25.38	7.89	2.03	$24.23_{(1.15)}$	$7.14_{(0.75)}$	2.04
Inception-ResNet-v2 [42]	$19.9^{\dagger}$	$4.9^{\dagger}$	20.37	5.21	11.75	$19.80_{(0.57)}$	$4.79_{(0.42)}$	11.76

ILSVRC 2017 Classification Competition

	224 × 224		320 × 320 / 299 × 299	
	top-1 err.	top-5 err.	top-1 err.	top-5 err.
ResNet-152 [10]	23.0	6.7	21.3	5.5
ResNet-200 [11]	21.7	5.8	20.1	4.8
Inception-v3 [44]	-	-	21.2	5.6
Inception-v4 [42]	-	-	20.0	5.0
Inception-ResNet-v2 [42]	-	-	19.9	4.9
ResNeXt-101 (64 × 4d) [47]	20.4	5.3	19.1	4.4
DenseNet-264 [14]	22.15	6.12	-	-
Attention-92 [46]	-	-	19.5	4.8
Very Deep PolyNet [51] †	-	-	18.71	4.25
PyramidNet-200 [8]	20.1	5.4	19.2	4.7
DPN-131 [5]	19.93	5.12	18.55	4.16
SENet-154	**18.68**	**4.47**	**17.28**	**3.79**
NASNet-A (6@4032) [55] †	-	-	17.3‡	3.8‡
SENet-154 (post-challenge)	-	-	**16.88‡**	**3.58‡**

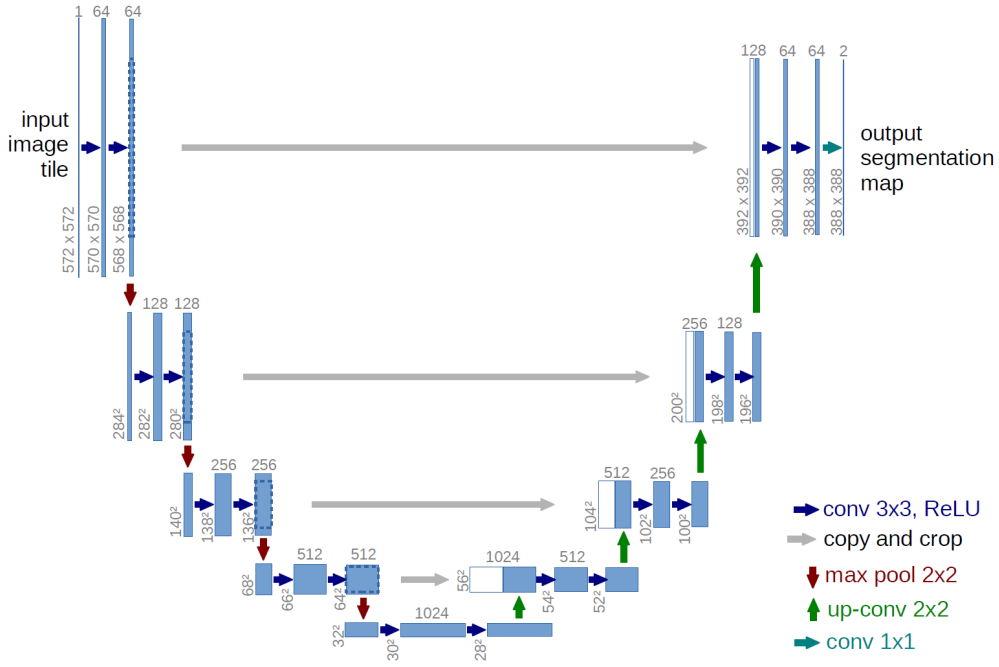
Single-crop error rates (%) on Places365 validation set

	top-1 err.	top-5 err.
Places-365-CNN [37]	41.07	11.48
ResNet-152 (ours)	41.15	11.61
SE-ResNet-152	**40.37**	**11.01**

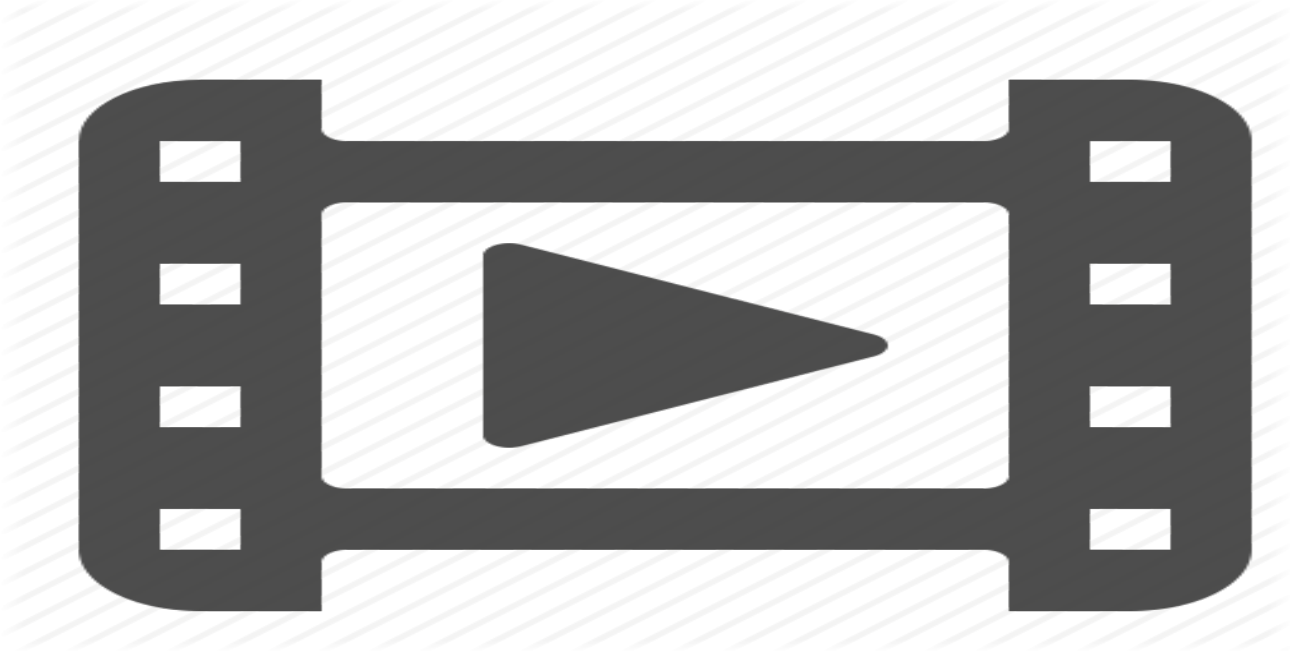
Object detection results on the COCO 40k validation set by using the basic Faster R-CNN

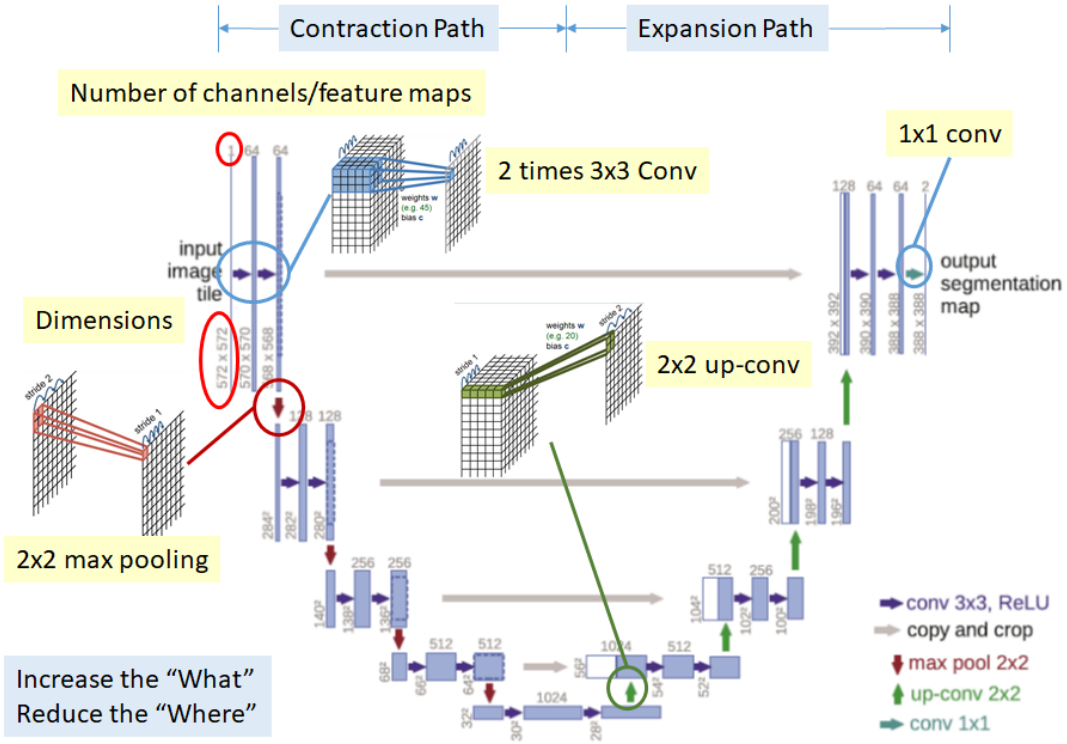
	AP@IoU=0.5	AP
ResNet-50	45.2	25.1
SE-ResNet-50	46.8	26.4
ResNet-101	48.4	27.2
SE-ResNet-101	49.2	27.9

U-Net [Ronneberger *et al.*, 2015]



https://towardsdatascience.com/review-u-net-biomedical-image-segmentation-d02bf06ca760unet

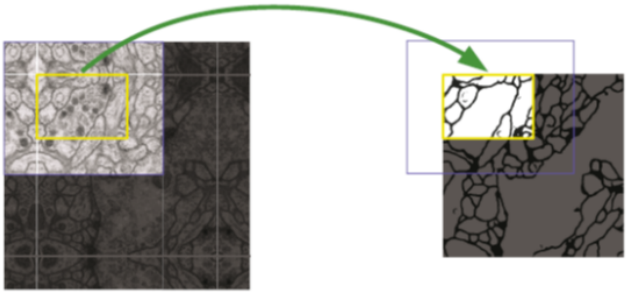



Contraction Path
Expansion Path
Number of channels/feature maps
1x1 conv
2 times 3x3 Conv
input image tile
output segmentation map
Dimensions
2x2 up-conv
2x2 max pooling
Increase the "What"
Reduce the "Where"
conv 3x3, ReLU
copy and crop
max pool 2x2
up-conv 2x2
conv 1x1

- **Contraction path**
  - Consecutive of two times of 3×3 Conv and 2×2 max pooling is done.
  - This can help to extract more advanced features but it also reduce the size of feature maps.

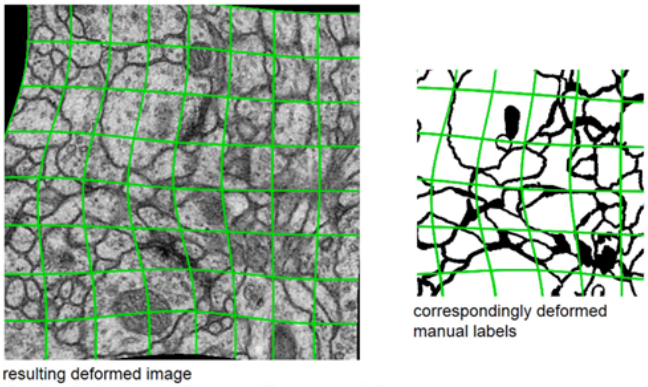
- **Expansion path**
  - Consecutive of 2×2 Up-conv and two times of 3×3 Conv is done to recover the size of segmentation map. However, the above process reduces the "where" though it increases the "what". That means, we can get advanced features, but we also loss the localization information.
  - Thus, after each up-conv, we also have concatenation of feature maps (gray arrows) that are with the same level. This helps to give the localization information from contraction path to expansion path.
  - At the end, 1×1 conv to map the feature map size from 64 to 2 since the output feature map only have 2 classes, cell and membrane.

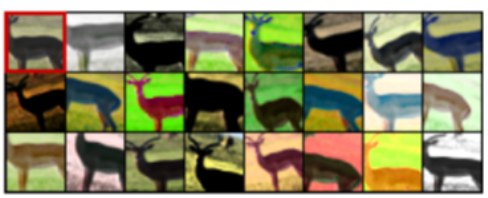
Overlap Tile Strategy



- Instead of downsizing before network and upsampling after network, overlap tile strategy is used.
- The whole image is predicted part by part.
- The yellow area in the image is predicted using the blue area.
- At the image boundary, image is extrapolated by mirroring.

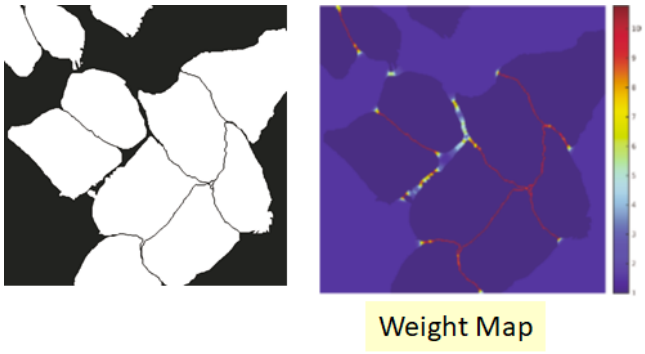
Elastic Deformation for Data Augmentation





From [3]

Separation of Touching Objects



Segmentation Map (Left) and Weight Map (Right)

- touching objects: closely placed each other, easily merged by the network
- applying a weight map to the output of network for separation

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \exp\left[-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right]$$

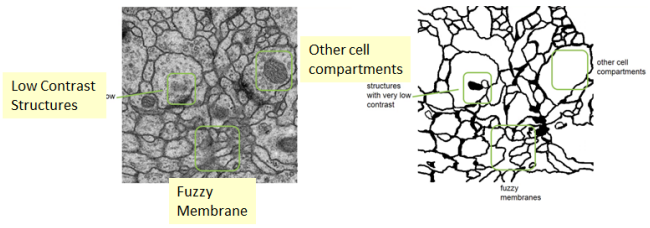
$d_1(\mathbf{x})$: the distance to the nearest cell border at position $\mathbf{x}$

$d_2(\mathbf{x})$: the distance to the second nearest cell border

$$p_k(\mathbf{x}) = \frac{\exp[a_k(\mathbf{x})]}{\sum_{l=1}^{K} \exp[a_l(\mathbf{x})]} \qquad E = \sum_{\mathbf{x}\in\Omega} w(\mathbf{x}) \log\left(p_{\ell(\mathbf{x})}(\mathbf{x})\right)$$

- The cross entropy function is penalized at each position by the weight map.
- And it help to force the network to learn the small separation borders between touching cells.

ISBI2012 Challenge

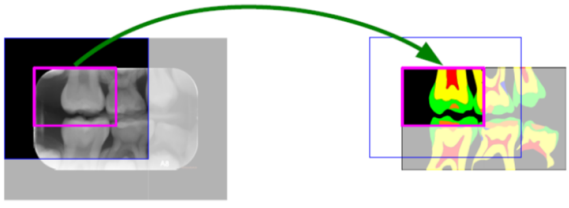


Some Difficult Parts in EM Images

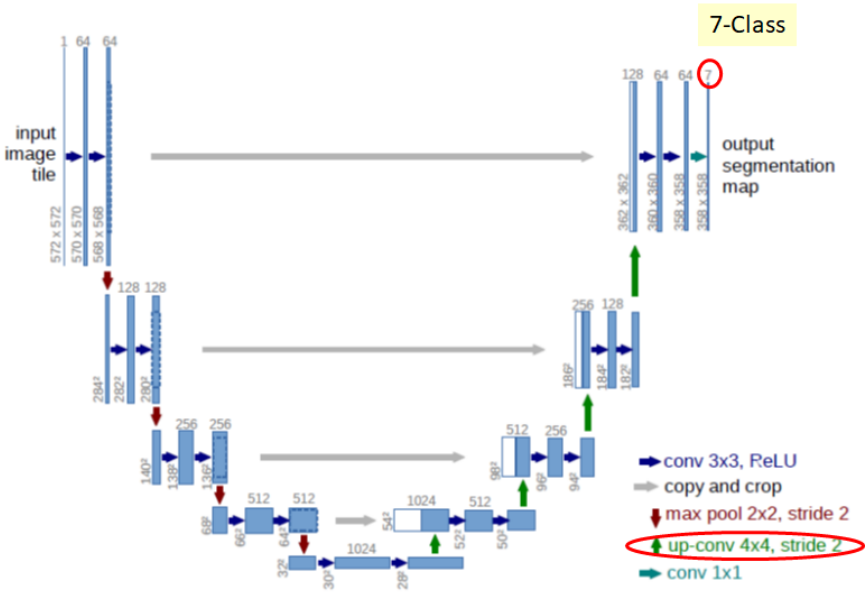
Rank	Group name	Warping Error	Rand Error	Pixel Error
	** human values **	0.000005	0.0021	0.0010
1.	u-net	**0.000353**	0.0382	0.0611
2.	DIVE-SCI	0.000355	0.0305	0.0584
3.	IDSIA [2]	0.000420	0.0504	0.0613
4.	DIVE	0.000430	0.0545	**0.0582**
⋮				
10.	IDSIA-SCI	0.000653	**0.0189**	0.1027

- Warping Error: A segmentation metric that penalizes topological disagreements.
- Rand Error: A measure of similarity between two clusters or segmentations.
- Pixel Error: A standard pixel-wise error.
- Training Hour: 10 Hours
- Testing speed: around 1s per image

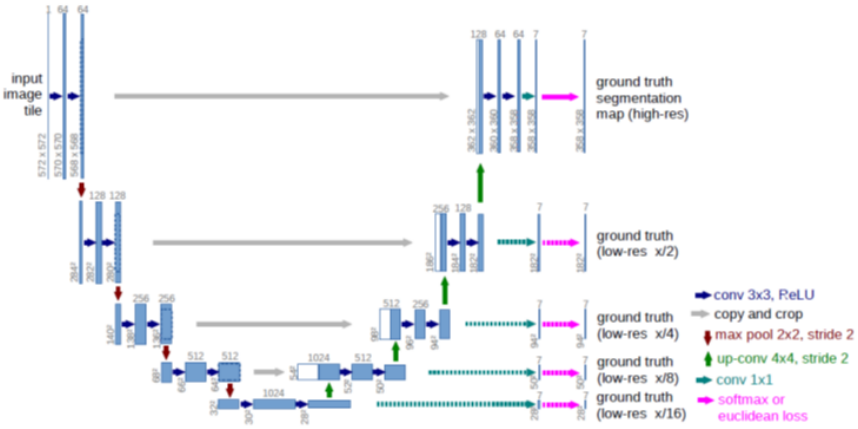
Dental X-Ray image with 7 classes



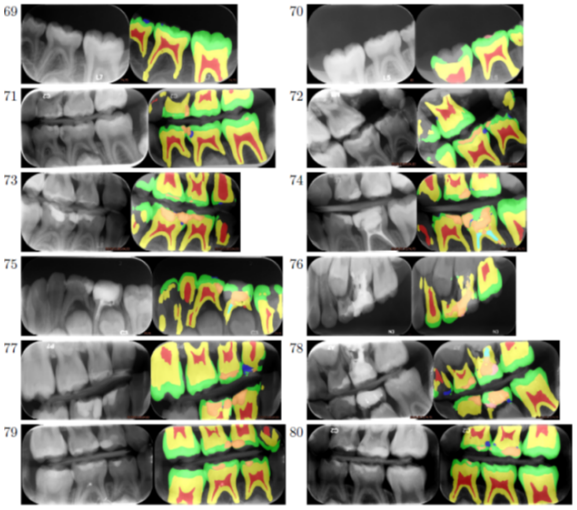
Zero padding instead of mirroring at the image boundary



Modified U-Net



Loss function at multiple levels



Visualization results

References

- Krizhevsky *et al.*, "ImageNet Classification with Deep Convolutional Neural Networks," NIPS, 2014
- Szegedy *et al.*, "Going Deeper with Convolutions," CVPR, 2015
- Simonyan *et al.*, "Very Deep Convolutional Networks for Large-Scale Image Recognition," ICLR, 2015
- Szegedy *et al.*, "Rethinking the Inception Architecture for Computer Vision," CVPR, 2016
- Szegedy *et al.*, "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," AAAI, 2017
- He *et al.*, "Deep Residual Learning for Image Recognition," CVPR, 2016
- Huang *et al.*, "Densely Connected Convolutional Networks," CVPR, 2017
- Hu *et al.*, "Squeeze-and-Excitation Networks," CVPR, 2018
- Ronneberger *et al.*, "U-Net: Convolutional Networks for Biomedical Image Segmentation," MICCAI, 2015

Thank you
for your attention!!!

(Q & A)

hisuk (AT) korea.ac.kr

http://milab.korea.ac.kr